Programming Languages

Peter Lo

Programming Language

- Set of words, symbols, and codes that enables a programmer to communicate instructions to a computer

The Translation Process

- Coding step translates a detail design representation of software into a programming language realization.
- Continues when a compiler accepts source code as input and produces machine-dependent object code as output.
- Compiler output is further translated into machine code, and these are the actual instructions that will actually be used by the central processing unit.

Programming Language Characteristics

- Coding process can be viewed firstly as communication via a programming language.
  - It is a human activity
  - Attention must be paid to the psychological characteristics of a language
- Coding process may also be viewed as one step in the software engineering process.
  - The engineering characteristics of a language therefore also has an important impact on the success of a software development project

Psychological Characteristics

- Uniformity
- Ambiguity
- Compactness
- Locality
- Linearity
- Tradition

Psychological Characteristics

- Uniformity
  - Indicates the degree to which a language uses consistent notation, applies arbitrary restrictions.
- Ambiguity
  - Refers to the situation where a programming language is perceived by the programmer in one way, but the compiler always interprets the language in another way.

Psychological Characteristics

- Compactness
  - Indicates the amount of code-oriented information that must be recalled from human memory.
- Locality
  - Measure of how much of a language that can be implemented as a "whole“.
  - Locality is enhanced when statements can be combined into blocks, and when design and resultant code are highly modular and cohesive.

Psychological Characteristics

- Linearity
  - A psychological characteristic that is closely associated with the concept of maintenance of functional domain, i.e. human perception is facilitated when a linear sequence of logical operations is encountered.
  - A programming language that does extensive branching violates the linearity of processing.
- Tradition
  - A programmer with experience in one form of language will find it easy to pick up another language that has the same sort of constructs in the former.

Engineering Characteristics

- Ease of design to code translation
- Compiler efficiency
- Source code portability
- Availability of development tools
- Maintainability of source code

Engineering Characteristics

- Ease of design to code translation
  - Indicates how closely a programming language can represent a design representation.
- Compiler efficiency
  - Many applications today still require fast, "tight" (i.e. low memory requirements) programs.
  - Languages with optimizing compilers may be required if software performance is a critical requirement.

Engineering Characteristics

- Source code portability
  - Whether source code may be transported from processor to processor and compiler to compiler with little or no modification.
- Availability of development tools
  - These can shorten the time required to generate source code and can improve the quality of code.
- Maintainability of source code
  - Source code should facilitate modifications according to any changes after implementation.

Factors should be considered in selecting a programming language

- General application area
- Algorithmic and computational complexity
- Environment in which software will execute
- Performance considerations
- Data structure complexity
- Knowledge of software development staff
- Availability of a good compiler or cross-compiler.

Programming Languages and Software Engineering

- Programming language will have an impact on project planning, analysis, design, coding, testing and maintenance.
- If complex data structures are required, languages with sophisticated data structure support (e.g. PASCAL) would be necessary.
- If high-performance, real-time capability is paramount, ADA would be appropriate. If memory-speed efficiency is in consideration, C would be more appropriate.
- In some instances, a complex data structure in design can only be satisfied by specific programming languages

Programming Languages Fundamentals

- **Data Types** and **Data Typing** can be described as a class of data objects together with a set of operations for creating and manipulating them.
- **Type Checking** is a mechanisms that:
  - Govern the operations that can be performed on a particular data type
  - And the manner in which different types can be manipulated in the same statement

Five Levels of Type Checking

- Level 0: Typeless
- Level 1: Automatic Type Coercion
- Level 2: Mixed Mode
- Level 3: Pseudostrong Type Checking
- Level 4: Strong Type Checking

Level 0: Typeless

- Programming languages have no explicit means for data typing and therefore, do not enforce type checking.

Level 1: Automatic Type Coercion

- Allows the programmer to mix different data types, but then converts operands of incompatible types, thus allowing requested operations to occur.

Level 2: Mixed-mode Type Conversion

- Different data types within the same type category are converted to a single target type so that a specified operation can occur.

Level 3: Pseudostrong Type Checking

- Similar to strong-type checking, but is implemented in a manner that provides one or more loopholes.

Level 4: Strong Type Checking

- Will only permit operations to be performed on data objects that are of the same data type.

Subprograms

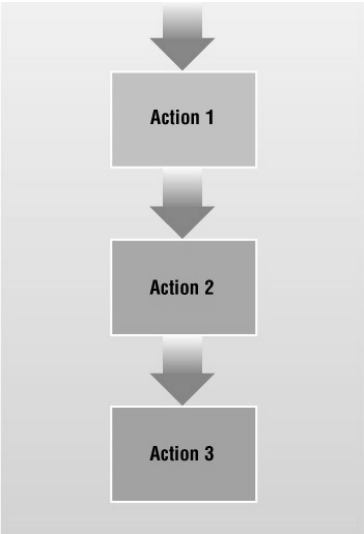
- A separately compatible program component that contains a data and control structure.
- A subprogram exhibits a number of generic characteristics:
  - A specification section that includes its name and interface characteristics
  - An implementation section that includes data and control structures
  - An activation mechanism that enables the subprogram to be invoked from elsewhere in the program.

Control Structures

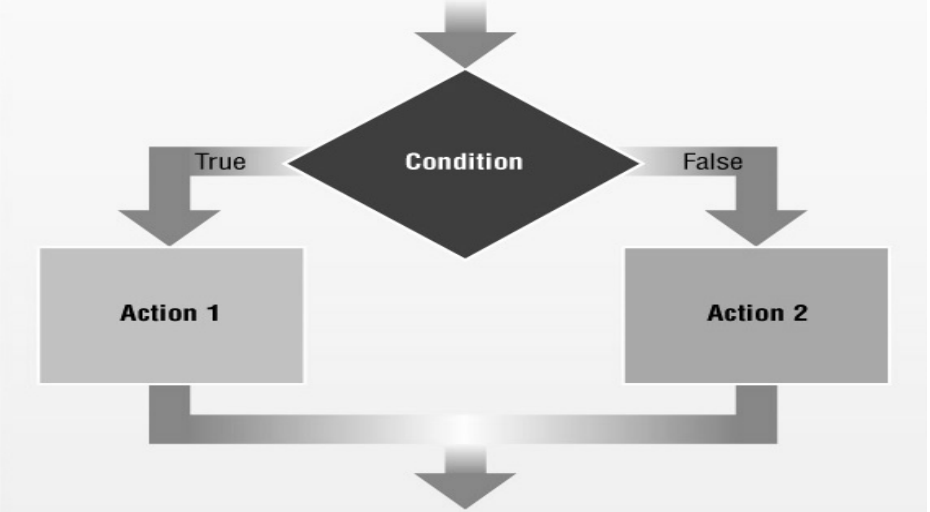
- All modern programming languages enable the programmer to represent sequence, condition, and repetition- the structured programming logical constructs.

Sequence Control Structure

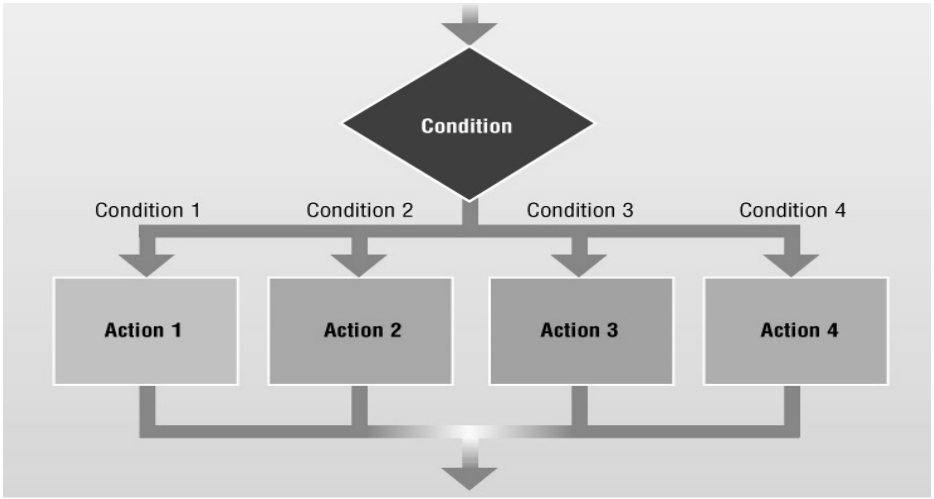
- Shows one or more actions following each other in order
- Actions could be
  - Inputs
  - Processes
  - Outputs



If-then-else Control Structure

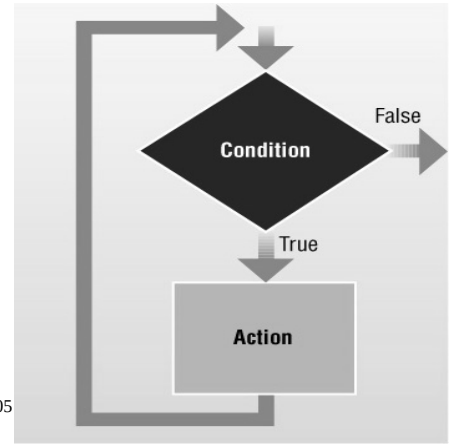


Case Control Structure



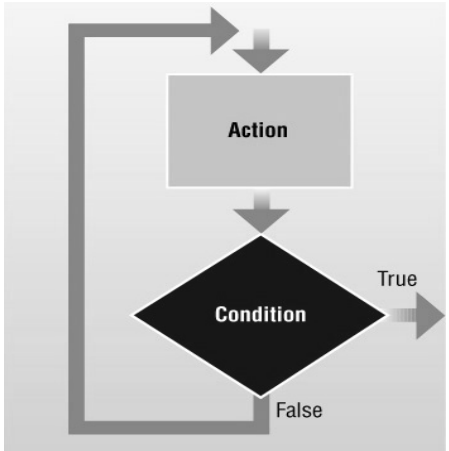
While Control Structure

- Repeats one or more times as long as condition is true



Do-While Control Structure

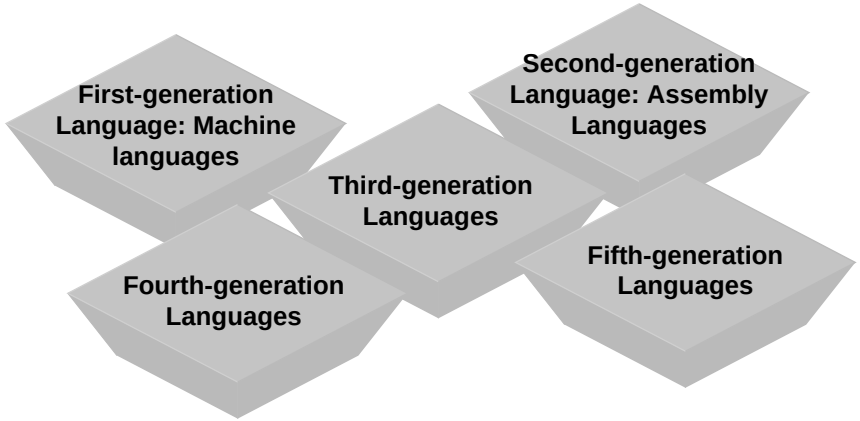
- Tests condition at end of loop



Low-level and High-level Programming Languages

low-level language	high-level language
Programming language that is machine-dependent	Language that is machine-independent
Machine-dependent language Runs only on one particular computer	Machine-independent language Can run on many different types of computers
Machine and assembly languages are low-level	Third-generation, fourth-generation, and fifth-generation languages are high-level

Categories of Programming Languages

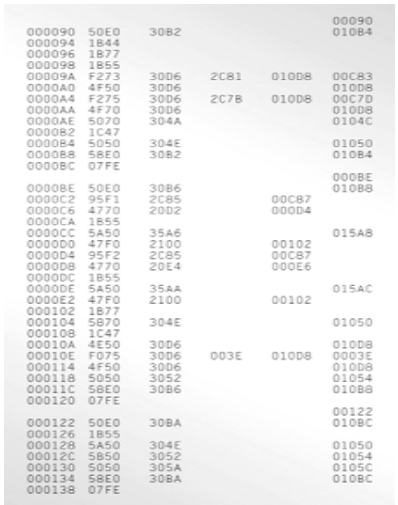


First Generation Languages

- The first language generation represents machine code and its more human-readable equivalent-assembly language.

Machine Language

- The only language that the computer directly understands.
- Functions as the object language of higher-level language programs, since all high-level languages must be translated into machine language in order for the computer to execute them.
- Often exists as octal or hexadecimal codes; extremely tedious to code in 0s and 1s.



Advantages and Disadvantages of Machine Language

- Advantage
  - Most efficient in terms of storage area use and execution speed.
  - Allows programmer to utilize the computer's potential for processing data.
- Disadvantage
  - Extremely difficult to program, remember and use.

Assembly Language

- Programmer uses symbolic names, or mnemonics, to specify machine codes.
- Mnemonics are English-like abbreviations for the machine-language opcodes.

```
*        THIS MODULE CALCULATES THE REGULAR TIME PAY
CALCSTPY EQU   *
         ST    14,SAVERTPY
         SR    4,4
         SR    7,7
         SR    5,5
         PACK  DOUBLE,RTHRSIN
         CVB   4,DOUBLE
         PACK  DOUBLE,RATEIN
         CVB   7,DOUBLE
         ST    7,RATE
         MR    4,7
         ST    5,RTPAY
         L     14,SAVERTPY
         BR    14
*        THIS MODULE CALCULATES THE OVERTIME PAY
CALCOTPY EQU   *
         ST    14,SAVEOTPY
TEST1    CLI   CODEIN,C'O'
         BH    TEST2
         SR    5,5
         A     5,=F'0'
         ST    5,OTPAY
         B     AROUND
TEST2    SR    4,4
         SR    7,7
         SR    5,5
         PACK  DOUBLE,OTHRSIN
         CVB   4,DOUBLE
         PACK  DOUBLE,RATEIN
         CVB   7,RATE
         MR    4,7
         MR    4,=F'1.5'
         ST    5,OTPAY
AROUND   L     14,SAVEOTPY
         BR    14
*        THIS MODULE CALCULATES THE GROSS PAY
CALCGPAY EQU   *
         ST    14,SAVEGPAY
         SR    5,5
         A     5,RTPAY
         A     5,OTPAY
         ST    5,GRPAY
         L     14,SAVEGPAY
         BR    14
```


Advantages and Disadvantages of Assembly Language

- Advantage
  - Can be used to develop programs highly efficient in terms of storage space use and processing time.
- Disadvantage
  - Cumbersome to use, as one assembly-language instruction is translated into one machine-language instruction.
  - Difficult to program effectively.
  - Machine-dependent, i.e. programs written on one computer generally cannot work on another.

Second Generation Languages

- These languages were developed in the late 1950's and the early 1960's, and served as the foundation for all third-generation languages.

Characteristic of Second Generation Languages

- Broad usage
- Enormous software libraries
- Widest familiarity and acceptance.
- Some examples of these are FORTRAN (30 years old), COBOL and BASIC.

Third Generation Languages

- Also called modern or structured programming languages.
- Three categories of 3GL:
  - General Purpose High-Order Languages
  - Object Oriented Languages
  - Specialized languages

General Purpose High-Order Languages

- Languages used for general programming purposes, e.g. software products, embedded applications and systems software.
- Examples: PASCAL, ALGOL, ADA and C.

Object Oriented Languages

- Object-oriented programming languages enable a software engineer to implement analysis and design models created using Object-oriented analysis and object-oriented design.
- Examples: dialects of C, i.e. C++, and Smalltalk.

Specialized Languages

- Characterized by unusual syntactic forms that have been especially designed for a distinct applications.
- Examples: LISP, PROLOG, APL and FORTH.

Fourth Generation Languages

- Combine procedure and non-procedure languages
- Enables the user to specify conditions and corresponding actions (the procedural component);
- While at the same time encouraging the user to indicate the desired outcome (the nonprocedural component);
- Then applying its domain-specific knowledge to fill in the procedural details.

Three categories of 4GL

- Query Languages
- Program Generators
- Other Categories of 4GLs

Query Languages

- Vast majority of 4GLs have been developed for use in conjunction with database applications.

Program Generators

- Program generators enable the user to create complete third-generation language programs.
- Most program generators today focus extensively on business information systems applications and generate programs in COBOL.

Other Categories of 4GLs

- Some of these other categories are Prototyping languages have been developed to assist in the creation of prototypes and a means for data modeling.
- Formal specification languages can also be considered as a 4GL when such languages produce machine-executable software.